symmetry

MDPI

Article

ieHDDP: An Integrated Solution for Topology Discovery and Automatic In-Band Control Channel Establishment for Hybrid SDN Environments

Joaquin Alvarez-Horcajo †, Isaias Martinez-Yelmo *, Elisa Rojas †, Juan Antonio Carral † and David Carrascal †

Departamento de Automática, Universidad de Alcalá, Escuela Politécnica Superior. Pza. San Diego, s/n, Alcalá de Henares, 28801 Madrid, Spain; j.alvarez@uah.es (J.A.-H.); elisa.rojas@uah.es (E.R.); juanantonio.carral@uah.es (J.A.C.); david.carrascal@uah.es (D.C.)

* Correspondence: isaias.martinezy@uah.es

† These authors contributed equally to this work.

Abstract: In-Band enhanced Hybrid Domain Discovery Protocol (ieHDDP) is a novel integral approach for hybrid Software-Defined Networking (SDN) environments that simultaneously provides a topology discovery service and an autonomous control channel configuration in the band. This contribution is particularly relevant since, to the best of our knowledge, it is the first all-in-one proposal for SDN capable of collecting the entire topology information (type of devices, links, etc.) and establishing in-band control channels at once in hybrid SDN environments (composed by SDN/no-SDN, wired/wireless devices), even with isolated SDN devices. ieHDDP facilitates the integration of heterogeneous networks, for example, in 5G/6G scenarios, and the deployment of SDN devices by using a simple exploration mechanism to gather all the required topological information and learn the necessary routes between the control and data planes at the same time. ieHDDP has been implemented in a well-known SDN software switch and evaluated in a comprehensive set of randomized topologies, acknowledging that ieHDDP is scalable in representative scenarios.

Keywords: hybrid; SDN; in-band control channel; topology discovery; OpenFlow

Citation: Alvarez-Horcajo, J.; Martinez-Yelmo, I.; Rojas, E.; Carral, J.A.; Carrascal, D. ieHDDP: An Integrated Solution for Topology Discovery and Automatic In-Band Control Channel Establishment for Hybrid SDN Environments. *Symmetry* **2022**, *14*, 756. https://doi.org/10.3390/sym14040756

Academic Editor: Chin-Ling Chen

Received: 16 March 2022
Accepted: 2 April 2022
Published: 6 April 2022

1. Introduction

The SDN paradigm is being embraced by the networking community due to the capabilities and features given by a programmable control plane [1]. Some of the main pillars of SDN are: (1) the separation of control and data planes, connected by a control channel; (2) the full knowledge of the underlying network topology and data plane; (3) the possibility to manage and set up the data plane at any time, based on the desired requirements and features of the control plane.

Concerning the first aspect, the SDN control channel is either based on dedicated connections that make up an out-of-band control channel or shared connections with regular data traffic, which is typically known as an in-band control channel. In-band control channels can be manually configured or based on auto-configuration mechanisms such as Amaru [2]. Regarding the knowledge of the underlying network topology, it is mostly based on protocols that convey the topological information from the data plane, such as the popular Open Flow Discovery Protocol (OFDP) [3]. Additionally, recent proposals also obtain complementary information, e.g., Tree Exploration Discovery Protocol (TEDP) [4] not only collects the topological information, but it also gathers routing information. Finally, the management of the data plane also requires protocols that allow establishing the desired rules to properly manage network traffic. These protocols can be OpenFlow or P4Runtime API (P4Runtime). However, all previously mentioned protocols are strictly applicable to fully-SDN environments. For this reason, alternatives such as enhanced Hybrid Domain

Discovery Protocol (eHDDP) [5] recently appeared to support hybrid SDN scenarios as a direct consequence of the inclusion of wireless and other heterogeneous network devices in the field. Nevertheless, the setup of SDN in-band control channels in hybrid SDN topologies still remains an open issue. In particular, the main challenge is how to integrate the capability to forward in-band control packets in non-SDN devices.

This article presents extended capabilities to our previous works Hybrid Domain Discovery Protocol (HDDP) [6], which is only suitable for wired networks, and eHDDP [5], which is suitable for hybrid wired/wireless networks, not only to convey topological information in hybrid SDN scenarios but also to establish SDN in-band control channels. Therefore, the proposal in this manuscript, ieHDDP, is capable of providing an integrated solution that allows effortless and plug-and-play use of SDN technology in any kind of scenario. This novel solution has been named ieHDDP and breaks with the traditional assumption in which the data plane is absolutely dependent on the control plane. This disruption is motivated by the symmetry introduced by ieHDDP that consists of a balanced coordination between the data and control planes. The data plane requires the control plane to have an adequate configuration, while the control plane requires the data plane to obtain the in-band control channel, which is key in hybrid heterogeneous networks where an out-band control channel for all devices is not feasible. Hence, ieHDDP fills in the gap by finding an integrated solution that performs both tasks (topology discovery and in-band control channel setup) at the same time in hybrid environments SDN. To the best of our knowledge, no work has merged both the topology discovery and in-band control channel configuration for generalized networks (that is, networks comprised of both wired and wireless links, and both SDN and non-SDN devices). Therefore, the main contributions of this paper are:

- The extension of eHDDP [5] (which already provides a topology discovery service) into ieHDDP to also establish in-band control channels.
- The implementation of the data plane part of ieHDDP in Basic OpenFlow User-space Software Switch (BOFUSS) [7] for wired devices.
- The implementation of the control plane part of ieHDDP in Open Network Operating System (ONOS).
- The evaluation of ieHDDP in wired hybrid SDN networks.

The paper is structured as follows. Section 2 presents the related work. Section 3 presents the necessary add-ons in ieHDDP to allow the exchange of SDN in-band control traffic between the hybrid SDN control and data planes. Section 4 presents the most relevant implementation details of the proposal, and Section 5 evaluates the proposal itself. Finally, Section 6 summarizes the conclusions of the presented work.

2. Related Work

As mentioned above, one of the pillars of the SDN paradigm is the knowledge of the underlying network of the data plane. This feature is covered for SDN-only networks by OFDP [3] or advanced solutions such as TEDP [4], eTDP [8]. Jia et al. [9] focused on improving its scalability and performance, which is a key issue in the field [10]. In the case of hybrid SDN networks, in which SDN and other devices coexist, some works try to infer the existence and connectivity of legacy devices (devices before the definition of the SDN architecture with distributed functionalities). On the one hand, the work in [11] combines Link Layer Discovery Protocol (LLDP) and Broadcast Domain Discovery Protocol (BDDP) to discover legacy devices in hybrid SDN networks, but it does not work properly in topologies containing loops of non-SDN devices. On the other hand, the work in [12] uses Forwarding and Control Element Separation (ForCES) [13] to detect non-SDN neighbor devices, but the proposal does not scale, since its passive mechanism based on LLDP and Address Resolution Protocol (ARP) cannot deal with topologies containing loops of non-SDN devices. Furthermore, the work in [14] attempts to solve the problem of link failures in hybrid scenarios, with coexisting SDN and legacy devices, applying machine learning-based policies. In this regard, the solution provided by eHDDP [5] allows the full

discovery of hybrid SDN topologies by using a specific protocol in non-SDN devices that conveys all the topological information to the SDN control plane even if loops of non-SDN devices exist.

At the same time, all the previous aforementioned solutions require an SDN control channel to transmit information from the data plane to the control plane. Although this channel might be deployed following either an out-of-band or in-band approach, the possibility of establishing an in-band control channel, even as a backup option (if the primary communication is out-of-band), significantly increases the resilience of SDN-based environments [15,16] and might provide additional functionality, such as inter-controller communication for distributed environments [17]. However, the dynamic establishment of in-band control channels requires certain autonomous functionality and/or the use of specific protocols to allow the correct exchange of SDN control traffic [18]. For example, some proposals rely on Rapid Spanning Tree Protocol (RSTP) to establish the SDN in-band control channel [19]. Additionally, reconfiguration after failure is also required and might be planned based on local re-routing [20]. Other approaches with global rerouting are as follows. Medieval [21] and FASIC [22] require manual configuration for the initial in-band control channel setup and control plane configuration to provide alternative paths in case of failures. FCCR [23] is a simulation-only study that creates the in-band control channel asynchronously among SDN devices and the recovery mechanism for failures is based on a complex mechanism based on failure detection in subrings. Izzy [24] is also a simulation-only study that requires a certain initial manual configuration and requires an AODV-like mechanism to establish alternative control paths if failures occur. ConForm [25] and Sakic et al. [19] build spanning tree-like in-band control channels rooted in the controller; the latter also establishes alternative paths from the control plane. Finally, Amaru [2] defines a novel discovery mechanism initiated by the control plane to obtain multiple paths to reach the control plane, which allows the required redundancy in case of failures.

Finally, few works emphasize the possibility of leveraging a global service for both topology discovery and in-band control bootstrapping, such as rXstp [26], and even fewer focus on the specific case of in-band control for wireless SDN [27]. Furthermore, the establishment of in-band control channels is only the first step to properly make use of the data plane. In this regard, additional considerations might be necessary for an optimized data plane when in-band control is leveraged, such as the installation of SDN rules and correct management of control traffic [28]. These considerations are complementary to the proposed work and, therefore, not directly related with the topic covered in this manuscript.

The final objective of the related work presented in this section is to provide the required functionality and the enabler technologies to integrate hybrid SDN scenarios that make feasible use cases such as Internet-of-Things (IoT) mobile-edge computing with enhanced capabilities, such as optimized task offloading [29], in which different network technologies coexist. Our ieHDDP proposal goes a step further in the state of the art toward the full-fledged integration of the SDN and IoT networking technologies, among other examples.

3. Support of SDN In-Band Signaling with ieHDDP

3.1. Problem Statement

In order to support SDN in-band control, a bidirectional connection between the control plane and an SDN device must be set up across the data plane. Since we consider eHDDP the starting point for the design ieHDDP, let us first describe its foundations. The eHDDP protocol [5] was designed to allow full topology discovery in hybrid networks, made of SDN and non-SDN devices. The protocol is triggered by the control plane and works in two phases. First, it explores the underlying network with a discovery message broadcast from the control plane, and then, it conveys the topological information acquired by the network devices during the exploration phase back to the control plane.

The original protocol, eHDDP, works on the basis that SDN devices already have an active connection (either in-band or outbound) to the control plane, so non-SDN devices

must only establish a path to reach an SDN device and from there the control plane, but this no longer holds with ieHDDP. In this new scenario, SDN devices are stand-by nodes; they cannot connect to the control plane because they lack the necessary knowledge (the *Controller Port* and, probably, even the control plane network address) until they are awakened by the discovery service exploration message. However, implementing the former eHDDP in SDN devices would allow them to learn a path to the controller by establishing the *Controller Port* to reach the control plane. Unfortunately, this is not enough to set up an in-band control channel, since eHDDP by itself does not provide support for the communication in the opposite direction, from the control plane to the network devices. Hence, our first and main goal is to provide a way so that those devices in the path from a given device toward the control plane (the chain of devices resulting from going back from device to device through their corresponding *Controller Port* until the controller is reached) learn the port pointing back to that specific device, thus conforming a bidirectional path that supports the in-band channel.

Moreover, taking into account that SDN devices are activated after receiving the discovery message, it is advisable that this message conveys certain information regarding the control plane, such as the IP address, port, or transport protocol. This information can also be preconfigured at devices as a primary control plane address, but its inclusion in the discovery message ensures that more than one controller can coexist in the network, and we grant additional flexibility in case it should be necessary for its dynamical update. Thus, it is up to the device to decide to stick to its primary control plane address or to abide by the address conveyed by the received discovery message.

3.2. Proposed ieHDDP Solution

In order to deal with the problems mentioned above, we propose ieHDDP, an extension of eHDDP, which is able to combine the topology discovery and support for in-band operation in hybrid networks composed of both SDN and non-SDN devices.

The original protocol works in two phases: a *Network exploration phase* designed to discover all the devices followed by a *Confirmation phase* in which the devices report back to the control plane. Firstly, the SDN controller will trigger the topology discovery service (the exploration phase in eHDDP). At this point, only SDN devices directly connected to the controller will have a pre-existing channel and function as a gateway for the rest (no matter if SDN or non-SDN, their control channel will be still inactive). At each step of the standard topology discovery service, all SDN devices traversed by the discovery message will be *awakened*, and this will trigger the creation of the in-band control channel *back* to the source gateway and, hence, toward the SDN controller. At the same time, all devices will have a twofold action, since they will also propagate the topology discovery service message *forwards* to continue the exploration of network devices, as in eHDDP.

We propose two updates to the original exploration phase of eHDDP. On the one hand, a new field is added to the discovery message to convey the address of the control plane to devices, that is, to identify the controller that initiates the process. On the other hand, when an SDN device learns which is the *Controller Port* (the first port that receives a copy of the discovery message), it triggers the in-band channel setup process (awakes its SDN capabilities) defined by ieHDDP. Non-SDN devices would learn the address of the control plane and its corresponding forwarding port for later use in the in-band control channel.

The exploration phase is followed by the confirmation phase. Every device receiving a late copy of the discovery message (at other ports) would report back to the control plane with a reply message, thus announcing its presence in the network. The reply is sent through the *Controller Port* (the one that received the first copy of the discovery message) and will be forwarded toward the control plane by other devices on the path (once its own presence on the path is added to the reply). No changes are needed in this phase for the new protocol.

As we mentioned in the previous section, in that way, devices in the network are not only discovered by the control plane, but they also learn the path to reach it. However,

to set up the in-band channel, we also need a path in the opposite direction. To accomplish this new functionality, a new logic has been added to the protocol.

This new logic relies on the *ARP Request* and *Reply* packet exchange triggered by SDN devices to establish the connection with the control plane as soon as the *Controller Port* is made available by the ieHDDP agent (although the *ARP* process is leveraged in this case, as we consider a common IPv4 network, Neighbor Discovery Protocol (NDP) would also be similarly applicable for IPv6-based networks). This packet exchange occurs concurrently with the original eHDDP exploration and confirmation phases, but it can be easily explained as a process composed of two additional phases called the *In-band Channel Learning* phase and *In-band Channel Confirmation* phase.

During the *In-band Channel Learning* phase, the *ARP Request* packet (issued by the SDN devices to learn the control plane Media Access Control (MAC) address before any communication) is sent across the network. Any ieHDDP-enabled device will receive the *ARP Request* and will learn the MAC address of the sending SDN device. Eventually, the corresponding *ARP Reply* would be sent back from the control plane, following the path previously set up by the original *ARP Request*. This *ARP Reply* is processed in every intermediate device to learn the control plane MAC address (*In-band Channel Confirmation* phase). At this point, every device in the path from an SDN device and the control plane knows how to communicate with them in both directions, thus enabling the setup of the in-band control channel.

3.3. ieHDDP Behavior Example

Figure 1 illustrates the operation of ieHDDP in a simple network made of four combined SDN/ieHDDP devices (switches S1, S2, S5, and S6) and two ieHDDP-only devices (switches S3 and S4). Only S1 can autonomously connect to the control plane, while S2, S5, and S6 will set up an in-band control channel when enabled through the ieHDDP protocol. For simplicity, only wired devices and the ieHDDP wired mode are considered in the example, but a similar approach would also apply for bidirectional wireless links. Note that at the beginning, only S1 is connected to the control plane (depicted in blue in the figure), while S2, S5 &, and S6 have to wait for ieHDDP to finish before setting up their SDN connection (they are depicted in white at the beginning), and S3 and S4 (depicted in gray), which are not SDN nodes, will never turn into the *connected* state.

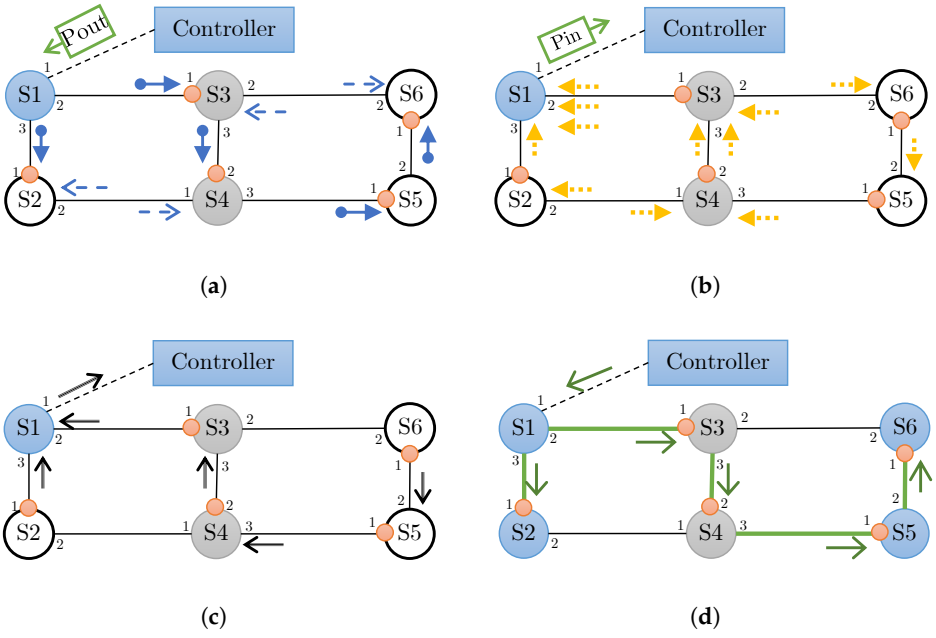


Figure 1. *Cont.*

Dev	If	Phase
Ctrl	1	Config
S5	2	c)
S6	2	c)
S2	3	c)

S1 Learning table

Dev	If	Phase
Ctrl	1	a)
S6	3	c)
S5	3	c)

S3 Learning table

Dev	If	Phase
Ctrl	2	a)
S6	3	c)
S5	3	c)

S4 Learning table

Dev	If	Phase
Ctrl	1	a)
S6	2	c)

S5 Learning table

(**e**)

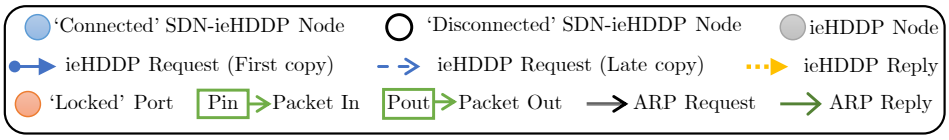


Figure 1. Example of In-Band Channel creation in ieHDDP. (**a**) Network exploration phase. (**b**) Confirmation phase. (**c**) In-band Channel Learning phase. (**d**) In-band Channel Confirmation phase. (**e**) ieHDDP learning tables.

3.3.1. Exploration Phase (Controller Port Learning)

The network exploration phase of ieHDDP (which is similar to the original phase in eHDDP) is shown in Figure 1a. It is triggered by the SDN control plane by flooding an *ieHDDP Request* message into the network through S1, as explained in [5]. The ingress port of the first copy received at each switch is *locked*, which prevents the switch from forwarding *late copies* of the same *ieHDDP Request* message received at other ports, to avoid loops. Then, the *locked* port is marked as *Controller port* in every node except for the ones already connected to the control plane (only S1 in our example), which also triggers the *In-band Channel Learning* phase.

3.3.2. Confirmation Phase (Topological Information Gathering)

This phase remains unchanged; it works exactly as in eHDDP (see Figure 1b). Whenever a *late copy* is received at a node, an *ieHDDP Reply* message is generated and sent back via the ingress port of the *late copy*. Each switch receiving an *ieHDDP Reply* message would update its contents to include itself on the route before relaying it toward the control plane via its *Controller Port*. Finally, when the *ieHDDP Reply* message arrives at S1, which is *connected* to the control plane, it is sent directly to it, via the corresponding *PacketIn* message, with no further processing.

3.3.3. In-Band Channel Learning Phase

Together with the next phase, the process shown in Figure 1c implements a basic learning switch in every combined SDN/ieHDDP device, so that a bidirectional path to the control plane can be learnt. It is triggered when the port locked by the first copy received from an *ieHDDP Request* message is marked as *Controller Port* at S2, S5, and S6. To set up the connection to the control plane, they send, only via their *Controller Port*, an *ARP Request* message looking for the MAC address of the control plane. Every intermediate node would process the *ARP Request* message and store the tuple <*source MAC address, ingress port*> into a *Learning Table*, before relaying the message, through its own *Controller Port*, toward the control plane. Finally, when the *ARP Request* message arrives at S1, it is processed in the same way and then sent directly to the server running the control plane. In our example, after the *ARP Request* messages from S2, S5, and S6 have been completely processed, a unidirectional path has been learned on the network, spanning from S1 to any of them. These paths would become bidirectional after the *In-band Channel Confirmation* and last phase.

3.3.4. In-Band Channel Confirmation Phase (Controller MAC Learning)

After receiving an *ARP Request* message, the server running the control plane issues the corresponding *ARP Reply*. S1 already knows how to reach the destination node of the *ARP Reply* (either S2, S5, or S6), they are stored into its *Learning Table* (see Figure 1d), so it simply forwards the *ARP Reply* to its destination. Now, the intermediate nodes on the path would process the *ARP Reply* and update their corresponding *Learning Table* with a new entry pointing to the control plane, the tuple <*source MAC address of the reply, ingress port*>, before relaying it to the next switch in the path. In this way, every switch visited by the *ARP Reply* would learn how to get to the control plane, thus transforming in a bidirectional manner the path to the replied destination switch. Finally, the bidirectional paths from S2, S5, and S6 to the control plane would be in place, and the in-band connections could be set up so that these switches become *connected* (they change from white to blue in Figure 1d); that means they can establish connections with the control plane as expected similarly to SDN devices, since the in-band control channel is already established and functional.

The control plane must periodically trigger all previous phases by executing the initial exploration phase. This periodicity guarantees both the detection of topology changes and reestablishment of the in-band control channel in case of failures. Furthermore, alternative in-band control paths may be established by the control plane, if necessary, as was proposed by other solutions previously mentioned in Section 2.

4. Implementation

Figure 2 shows the flow diagram of our proof-of-concept implementation, which is based on BOFUSS [7], a well-known SDN software switch.

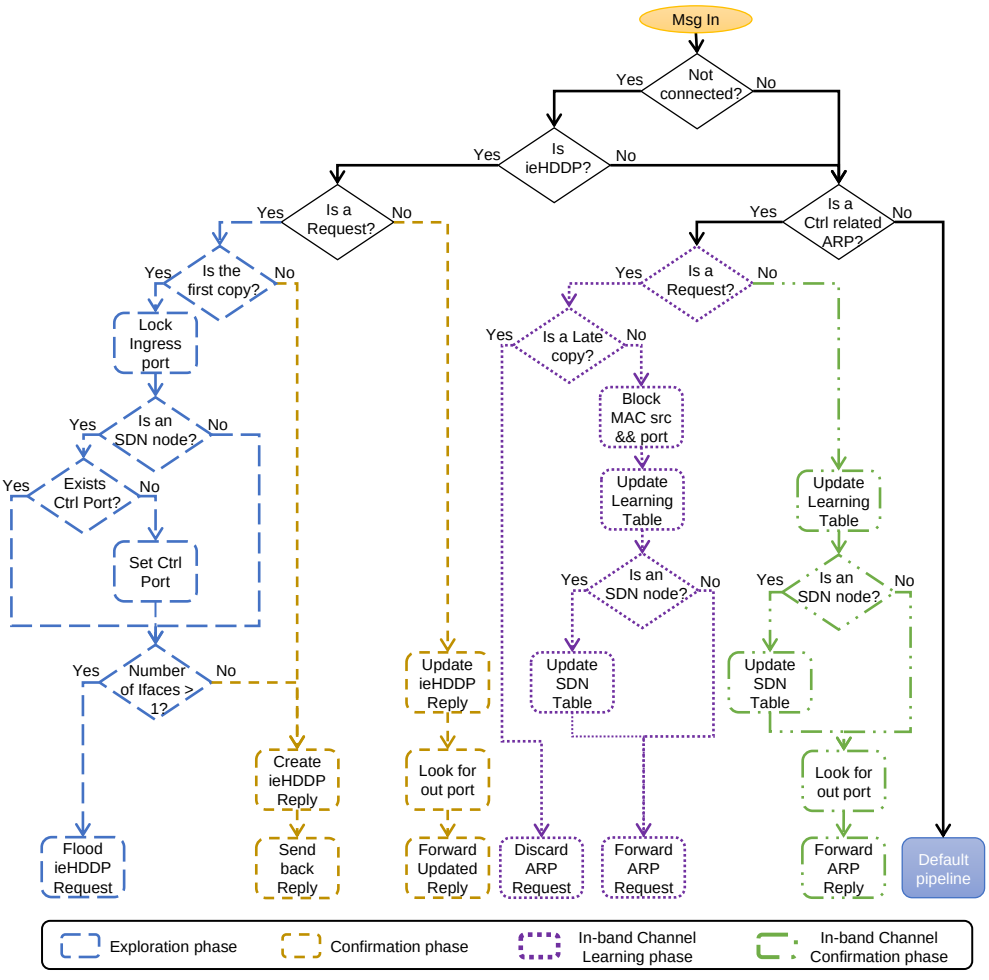


Figure 2. Implementation of the ieHDDP agent.

The flow chart is organized into four branches (depicted in different colors and patterns) that correspond to the four phases of the protocol, as explained in Section 3. When the switch is connected to the control plane and the message received is not an ARP related to the controller (either Request or Reply), the processing of incoming messages checks if it belongs to ieHDDP or ARP protocols to process its contents; otherwise, the processing is left to the default pipeline. Afterwards, two possibilities arise depending on whether they carry a Request or a Reply packet, so we end up with four processing options, i.e., the four branches just mentioned above.

The left branch of the flowchart, depicted in blue (and long dashed lines) in Figure 2, shows the processing related to the *Exploration phase* of ieHDDP. It is triggered by the reception of the first copy of an *ieHDDP Request* message. First of all, the agent locks the ingress port to avoid processing late copies of the same *ieHDDP Request* as an original request, which may produce loops. Then, the next operation depends on whether the switch also features an SDN agent. If true, it checks if the *Controller Port* has already been set (meaning an active connection to the control plane has been initiated and there is nothing else to do) and otherwise sets the *Controller Port* as the ingress port of the *ieHDDP Request*, which acts as a trigger to the SDN agent to initiate the connection. Finally, it checks whether it is a single interface; if true, it forces an immediate *ieHDDP Reply* back to the control plane (in our scheme, the flow moves to the next phase); otherwise, it broadcasts the *ieHDDP Request* to ensure that it reaches every switch in the network. This last check is compliant with eHDDP and allows to discover devices with only one interface sharing that link with their only neighbor.

The middle-left branch of the flowchart, depicted in yellow in (and dashed lines) Figure 2, shows the processing related to the *Confirmation phase* of ieHDDP. This phase manages the *ieHDDP Reply* messages, either to generate and send them to the control plane or to process the replies generated by other switches on their way to the control plane. Whenever a late copy of an *eHDDP Request* is received, an *ieHDDP Reply* is created and sent back through the ingress port of the *eHDDP Request*. The *ieHDDP Reply* carries the information regarding the switch *Id* and the ingress port of the *eHDDP Request*; in this way, it conveys the information about the link to the control plane. On the other hand, when an *ieHDDP Reply* is received, it updates its contents by also including the tuple <*switch Id, ingress port*> before forwarding it to the control plane through its *Controller Port*.

The right side of the flowchart covers the processing of *ARP Request* and *Reply* messages related to the Control Plane. The middle right branch of the flowchart, depicted in purple (and doted lines) in Figure 2, shows the processing related to the *In-band Channel Learning phase* of ieHDDP, which is triggered by the reception of an *ARP Request* message. This message is broadcast in the network so multiple copies of the same packet could be received at a given switch. When the first copy of the *ARP Request* is received, the switch locks the ingress port to avoid broadcast loops and updates its *Learning Table* by inserting a new entry pointing to the source MAC address of the *ARP Request*. If the switch features an SDN agent, it also sends a message to force the insertion of the corresponding rule in the SDN table. Finally, it forwards the *ARP Request* to the control plane. Otherwise, if the *ARP Request* received is a late copy of a previous request, it is simply discarded to avoid loops of broadcast traffic.

The last branch (and the rightmost one) of the flowchart, shown in green (and dash-doted lines) in Figure 2, shows the processing of incoming *ARP Reply* messages. The switch updates its *Learning Table* by inserting a new entry pointing to the source MAC address of the reply (i.e., the controller MAC address). If the switch features an SDN agent, it also sends a message to force the insertion of the corresponding rule into the SDN table. Finally, it looks for the destination port in its *Learning Table* and forwards the *ARP Reply*.

5. Evaluation

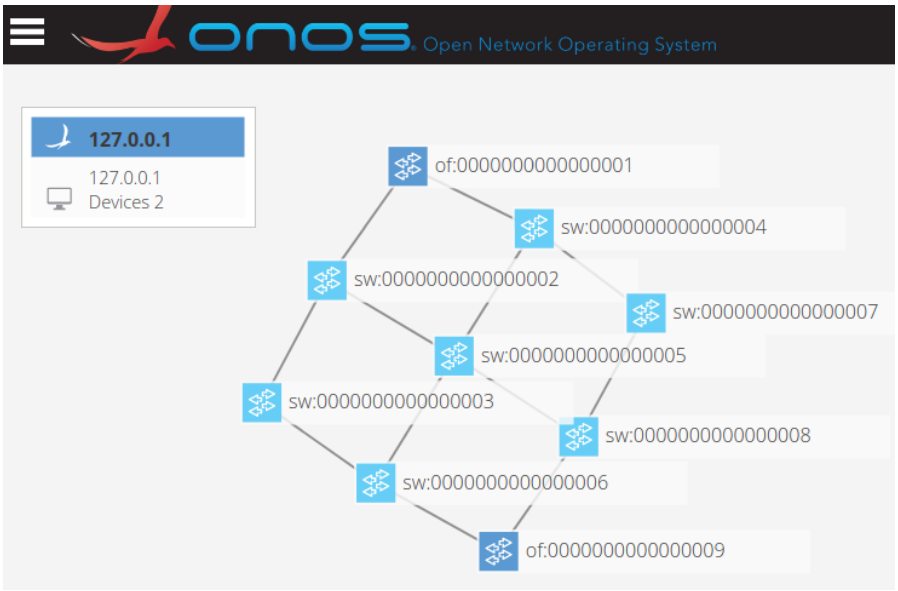
The evaluation of ieHDDP was performed on Intel(R) Core(TM) i7 CPUs (12 core) and 24 GB of RAM servers, running ONOS as the SDN controller as well as the emulated topologies. The network devices are deployed in independent Linux network namespaces connected through virtual Ethernet interfaces. This setup allows the isolation of the emulated network devices with respect to their host to ensure the communication with the control plane through the in-band control channel.

The obtained results only show the performance of ieHDDP, since there is no other solution that provides topology discovery and in-band control channel autoconfiguration in hybrid SDN networks.

5.1. Proof-of-Concept

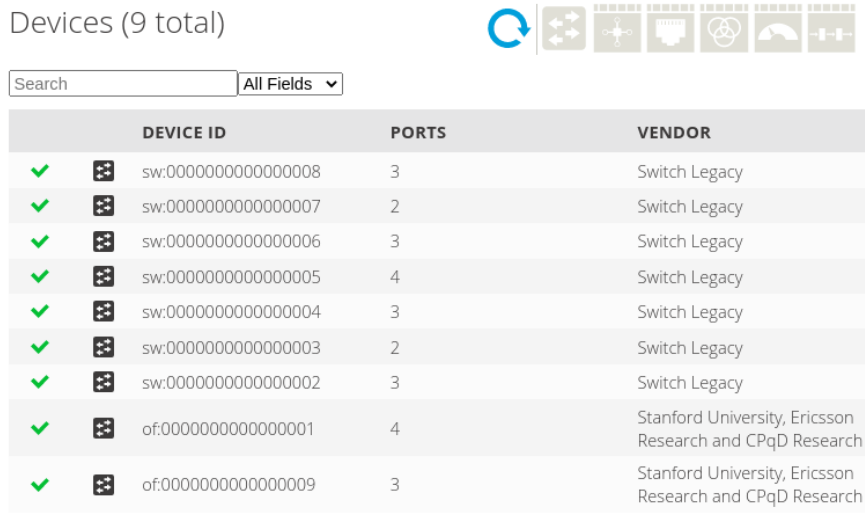
To verify the correct operation of the ieHDDP in-band selection process, a proof-of-concept has been performed. This test was conducted on a 3 × 3 square mesh topology composed of seven ieHDDP-only and two combined SDN-ieHDDP nodes. The controller is connected to one of the combined SDN-ieHDDP nodes, while the other combined node is located in the opposite corner of the mesh. We outlined this scenario as a worst-case situation because several non-SDN nodes must be traversed to set up the in-band control channel.

Figure 3a shows the 3 × 3 square mesh topology detected by the ONOS controller. The combined SDN-ieHDDP nodes are depicted in dark blue, while ieHDDP-only nodes are depicted in light blue. We can observe how the ieHDDP-only nodes are placed in the middle of the topology (switches S2 through S8), while the combined SDN-ieHDDP nodes are located at the opposite corners (switches S1 and S9). Therefore, with this proof-of-concept, we prove that the controller can create the topology graph as expected. Furthermore, it is demonstrated how the in-band control channel has been properly set up crossing multiple ieHDDP-only nodes. Moreover, Figure 3b illustrates the devices detected by the controller in more detail. The two devices with an ID consisting of *"of:"* followed by a *Datapath ID (DpID)* represent the SDN-ieHDDP nodes, while the remaining, labeled as *"sw:"* plus a *DpID*, are the ieHDDP-only nodes. Likewise, the number of ports of each device is shown as well as a green tick that indicates all nodes are operational. Finally, Figure 3c depicts additional information acquired by ONOS about both types of nodes.

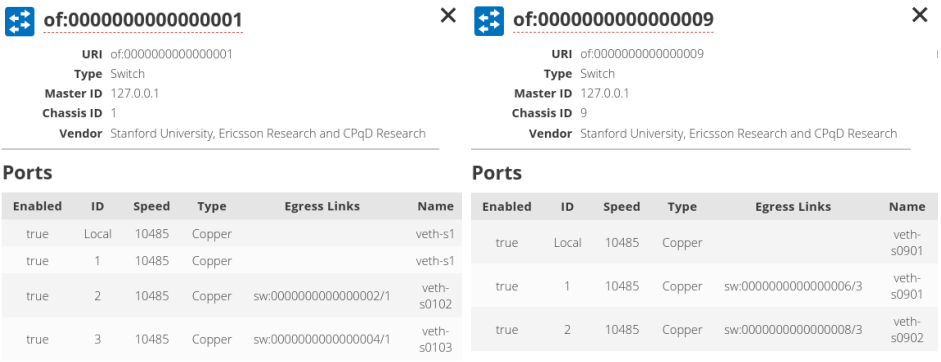


(**a**)

Figure 3. *Cont.*



(**b**)



(**c**)

Figure 3. Graphical interface of ONOS illustrating the topology and network devices. (**a**) Mesh hybrid topolody discovered. (**b**) Devices detected by the ONOS controller. (**c**) Info on SDN devices acquired by ONOS.

5.2. ieHDDP Protocol Performance

To assess the performance of our ieHDDP implementation, we designed and carried out several experiments on random topologies, following the Barabási-Albert [30] and Waxman [31] models of increasing size (10, 15, and 20 nodes). Only one node per topology is directly connected to the control plane (acting as the *gateway*), while the rest of the nodes will wait for the in-band control channel to be set up before connecting. Therefore, this is a worst-case scenario in terms of both the number of control packets exchanged and the time elapsed to establish the connection. For each topology size, three sets of 10 randomized topologies were generated using BRITE, corresponding to average topology connectivity degrees of 2, 4, and 6. Finally, each experiment was repeated at least 10 times to compute 95% confidence intervals by selecting a different node as a *gateway* in each run.

First, we measured the time required to perform the *Controller Port* selection (i.e., the exploration time in ieHDDP) and to complete the in-band control channel setup. The left part of Figure 4 shows the exploration time, measured as the time elapsed from the moment the first *ieHDDP Request* message is created in the control plane until it is received by all nodes in the network. We observe that it is in the range of a few hundreds of milliseconds and clearly increases with the network size (number of nodes), as the top graph shows 10 nodes and the bottom one shows 20. At the same time, regarding the network connectivity, we can see how the exploration time decreases with increasing

node degree (three bars with three different colors). This is due to the tree-like nature of the exploration process, producing trees that grow in breadth instead of in depth with increasing node degree; i.e., higher connectivity degrees accelerate the exploration process. Finally, regarding the network model, we can observe the influence of the hub-and-spoke nature of Barab'asi-Albert's model, which clearly reduces the exploration time (nodes are located at fewer hops to the gateway), especially for low-degree nodes.

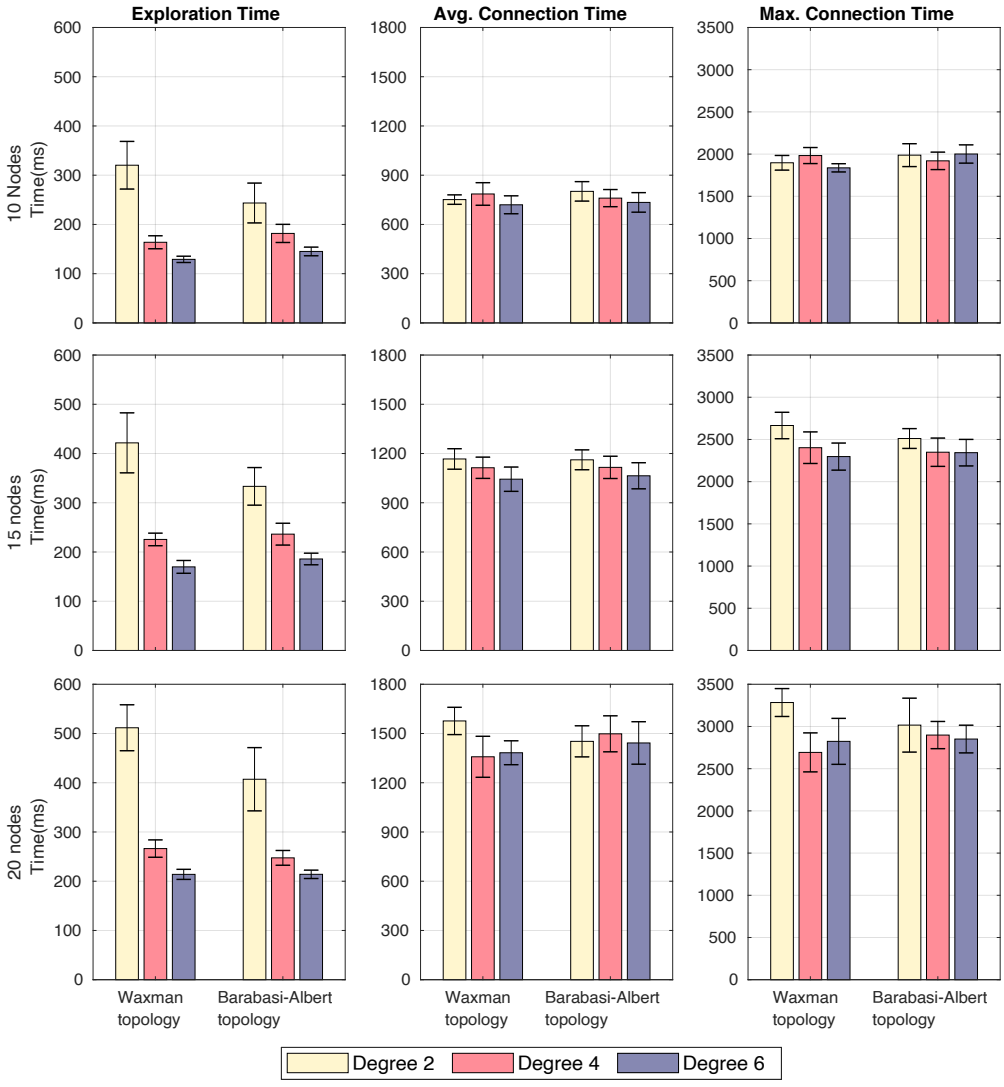


Figure 4. ieHDDP exploration and connection times in various topologies.

In addition, in Figure 4, the center and right columns depict the results regarding the connection time (both as an average and as maximum). The connection time is measured as the time elapsed from the creation of the first *ieHDDP Request* message on the controller until the OpenFlow connection between the controller and the node is up and running, that is, until the `HELLO`, `OFPT_FEATURES_REQUEST` and `OFPT_FEATURES_REPLY` messages have been exchanged. The average connection time ranges from around 600 ms to 1.5 s on average and up to 3 s as maximum. They are strongly dependent on the number of nodes because of two main factors: first, the processing times of protocol messages at nodes and, second, the availability of processor cores in the emulation servers. In the case of only 10 nodes, every switch process runs on a dedicated processor core (our hardware possesses 12 CPU cores), but with 15 and 20 nodes, the effect of core swapping between switch processes becomes non-negligible. Experiments with a larger number of nodes have been conducted, but their results are not included, since they are not representative due to

the aforementioned core swapping effect, which increases substantially as the number of considered nodes increases.

Moreover, Figure 5 illustrates the number of control packets (Request and Reply) used. It is important to notice that apart from the original message exchange in eHDDP, no additional control messages are needed for the in-band control setup. We can observe that both figures are proportional to the number of links in the topology, as expected.

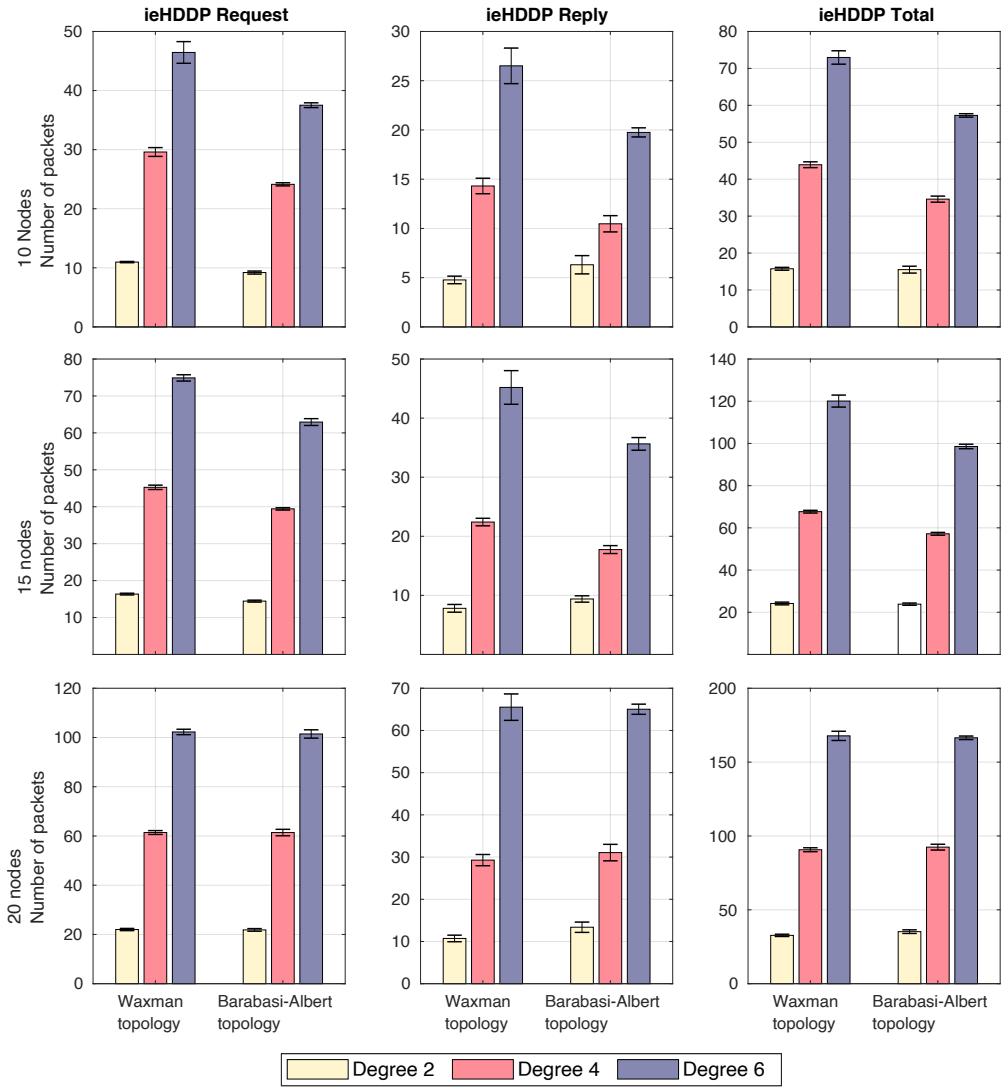


Figure 5. Number of exchanged ieHDDP control packets.

Finally, Figure 6 shows the number of ARP protocol (Request and Reply) packets issued by nodes to learn the MAC address of the controller, which ieHDDP leverages to learn the in-band path in the direction from the controller to the nodes. Again, these packets are not generated by our protocol; they are part of the usual exchange of the TCP/IPv4 connection exchange. ieHDDP simply reuses them to learn the path for the in-band control channel. In fact, ieHDDP not only benefits from it but also reduces the amount of *ARP Request packets*, because they are sent in unicast through the *Controller Port* discovered in the exploration phase, instead of flooding them as usual.

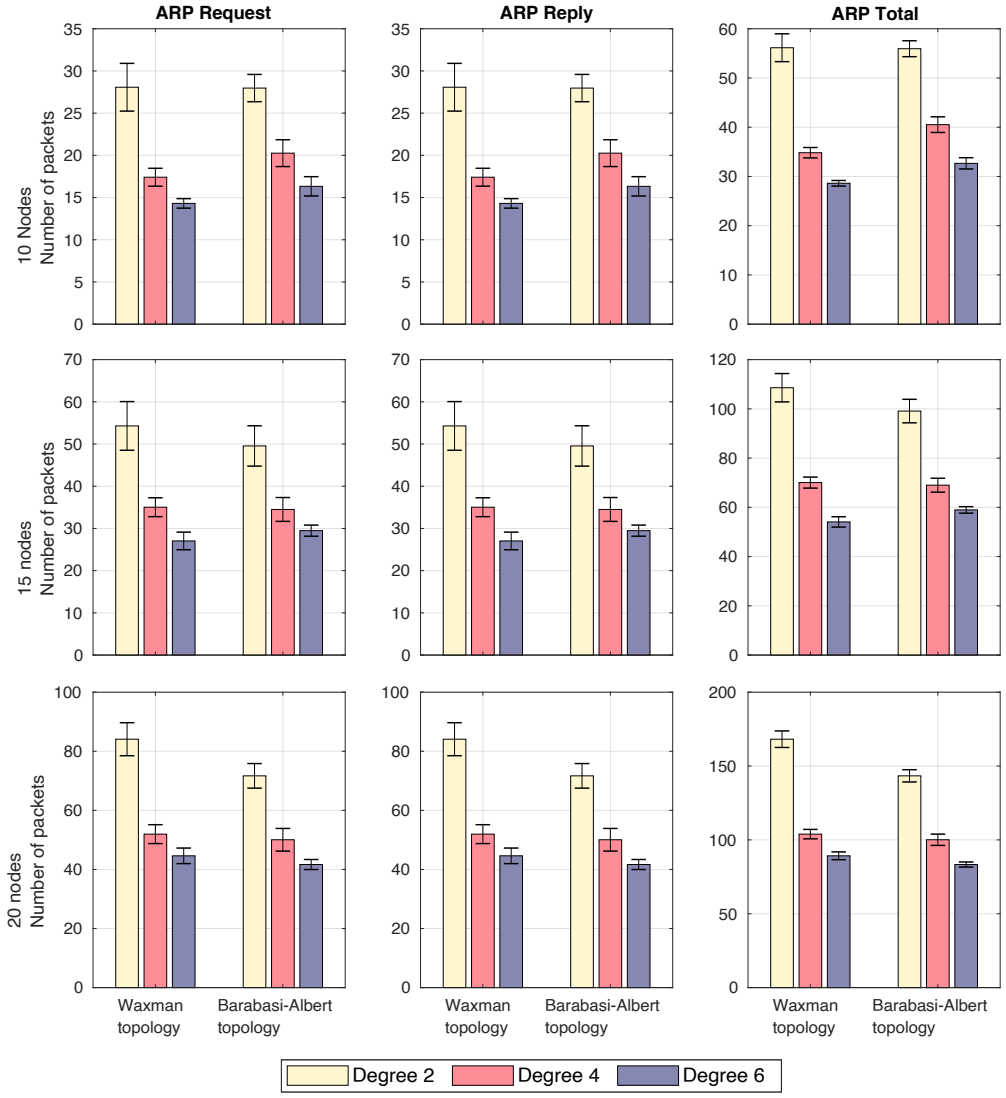


Figure 6. Number of ARP protocol packets exchanged for the in-band control channel set up.

6. Conclusions

This paper has presented ieHDDP, which is an enhancement of eHDDP that cannot only convey topological information but is also capable of establishing in-band control channels in hybrid SDN domains in which SDN/no-SDN and wired/wireless devices coexist. The in-band control channel establishment is based on an exploration process triggered by the control plane, which reaches all devices via a controlled flooding mechanism that allows discovering the port/next-hop to establish a path from each device to the SDN controller(s), and at the same time, it recollects all the required topological information. This path allows SDN devices to establish their connections with the SDN control plane.

The proposed mechanism to recollect the topological information (derived from eHDDP) is described in Sections 3.3.1 and 3.3.2, while the mechanism to establish—in parallel—the in-band control channel is detailed in Sections 3.3.3 and 3.3.4, by reusing the *ieHDDP Request* that explores the network from the control to the data plane.

The implementation detailed in Section 4 has been evaluated for wired scenarios in Section 5. The obtained results are promising since the required number of packets is linearly proportional to the existing number of links in the randomized topologies of the experiments, and exploration and connection times are also linearly proportional to the diameter of the network topology (it increases with the number of devices and decreases if the degree of the network increases), as expected. This behavior of ieHDDP grants scalability in large-scale network scenarios.

As future work, we plan to extend the experiments including wireless devices and design the required enhancements for wireless topologies with unidirectional—non-bidirectional—links, in which ieHDDP is not directly applicable to achieve a bidirectional path to the controller, as currently defined.

Author Contributions: Conceptualization, E.R. and I.M.-Y.; methodology, I.M.-Y. and J.A.C.; software, J.A.-H. and D.C.; validation, I.M.-Y. and J.A.-H.; formal analysis, E.R. and J.A.C.; investigation, J.A.-H. and D.C.; resources, J.A.C.; data curation, J.A.-H. and D.C.; writing—original draft preparation, I.M.-Y. and J.A.-H.; writing—review and editing, I.M.-Y., J.A.C. and E.R.; visualization, D.C. and J.A.C.; supervision, E.R. and I.M.-Y.; project administration, E.R. and I.M.-Y.; funding acquisition, E.R. and I.M.-Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by grants from Comunidad de Madrid through projects TAPIR-CM (S2018/TCS-4496), IRIS-CM (CM/JIN/2019-039) and MistLETOE-CM (CM/JIN/2021-006), and by project ONENESS (PID2020-116361RA-I00) of the Spanish Ministry of Science and Innovation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Source code at https://github.com/NETSERV-UAH/ieHDDP (accessed on 6 March 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACL	Access Control List
ARP	Address Resolution Protocol
BOFUSS	Basic OpenFlow User-space Software Switch
DpID	Datapath ID
eHDDP	enhanced Hybrid Domain Discovery Protocol
ForCES	Forwarding and Control Element Separation
HDDP	Hybrid Domain Discovery Protocol
ieHDDP	In-Band enhanced Hybrid Domain Discovery Protocol
IoT	Internet of Things
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
NDP	Neighbor Discovery Protocol
OFDP	Open Flow Discovery Protocol
ONOS	Open Network Operating System
P4R	P4Runtime API
RSTP	Rapid Spanning Tree Protocol
SDN	Software-Defined Networking
TEDP	Tree Exploration Discovery Protocol

References

1. Farhady, H.; Lee, H.; Nakao, A. Software-Defined Networking: A survey. *Comput. Netw.* **2015**, *81*, 79–95. [CrossRef]
2. Lopez-Pajares, D.; Alvarez-Horcajo, J.; Rojas, E.; Asadujjaman, A.S.M.; Martinez-Yelmo, I. Amaru: Plug Play Resilient In-Band Control for SDN. *IEEE Access* **2019**, *7*, 123202–123218. [CrossRef]
3. Pakzad, F.; Portmann, M.; Tan, W.L.; Indulska, J. Efficient Topology Discovery in Software Defined Networks. In Proceedings of the International Conference on Signal Processing and Communication Systems, Gold Coast, Queensland, Australia, 15–17 December 2014; pp. 1–8. [CrossRef]
4. Rojas, E.; Alvarez-Horcajo, J.; Martinez-Yelmo, I.; Carral, J.A.; Arco, J.M. TEDP: An Enhanced Topology Discovery Service for Software-Defined Networking. *IEEE Commun. Lett.* **2018**, *22*, 1540–1543. [CrossRef]
5. Martinez-Yelmo, I.; Alvarez-Horcajo, J.; Carral, J.A.; Lopez-Pajares, D. eHDDP: Enhanced Hybrid Domain Discovery Protocol for network topologies with both wired/wireless and SDN/non-SDN devices. *Comput. Netw.* **2021**, *191*, 107983. [CrossRef]
6. Alvarez-Horcajo, J.; Rojas, E.; Martinez-Yelmo, I.; Savi, M.; Lopez-Pajares, D. HDDP: Hybrid Domain Discovery Protocol for Heterogeneous Devices in SDN. *IEEE Commun. Lett.* **2020**, *24*, 1655–1659. [CrossRef]

7. Fernandes, E.L.; Rojas, E.; Alvarez-Horcajo, J.; Kis, Z.L.; Sanvito, D.; Bonelli, N.; Cascone, C.; Rothenberg, C.E. The road to BOFUSS: The basic OpenFlow userspace software switch. *J. Netw. Comput. Appl.* **2020**, *165*, 102685. [CrossRef]
8. Ochoa-Aday, L.; Cervelló-Pastor, C.; Fernández-Fernández, A. eTDP: Enhanced Topology Discovery Protocol for Software-Defined Networks. *IEEE Access* **2019**, *7*, 23471–23487. [CrossRef]
9. Jia, Y.; Xu, L.; Yang, Y.; Zhang, X. Lightweight Automatic Discovery Protocol for OpenFlow-Based Software Defined Networking. *IEEE Commun. Lett.* **2020**, *24*, 312–315. [CrossRef]
10. Wazirali, R.; Ahmad, R.; Alhiyari, S. SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Appl. Sci.* **2021**, *11*, 6999. [CrossRef]
11. Ochoa Aday, L.; Cervelló Pastor, C.; Fernández Fernández, A. *Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks*; Technical Report; Universitat Politécnica de Catalunya, Departament d'Enginyeria Telemática: Barcelona, Spain, 2015.
12. Tarnaras, G.; Athanasiou, F.; Denazis, S. Efficient Topology Discovery Algorithm for Software-Defined Networks. *IET Netw.* **2017**, *6*, 157–161. [CrossRef]
13. Yang, L.; Dantu, R.; Anderson, T.; Gopal, R. *Forwarding and Control Element Separation (ForCES) Framework*; Technical Report 3746; Internet Engineering Task Force: Fremont, CA, USA, 2004.
14. Ibrar, M.; Wang, L.; Muntean, G.M.; Akbar, A.; Shah, N.; Malik, K.R. PrePass-Flow: A Machine Learning based technique to minimize ACL policy violation due to links failure in hybrid SDN. *Comput. Netw.* **2021**, *184*, 107706. [CrossRef]
15. Huang, H.; Guo, S.; Liang, W.; Li, K.; Ye, B.; Zhuang, W. Near-Optimal Routing Protection for In-Band Software-Defined Heterogeneous Networks. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2918–2934. [CrossRef]
16. González, S.; De la Oliva, A.; Bernardos, C.J.; Contreras, L.M. Towards a Resilient Openflow Channel Through MPTCP. In Proceedings of the 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Valencia, Spain, 6–8 June 2018; pp. 1–5. [CrossRef]
17. Hark, R.; Rizk, A.; Richerzhagen, N.; Richerzhagen, B.; Steinmetz, R. Isolated in-band communication for distributed SDN controllers. In Proceedings of the 2017 IFIP Networking Conference (IFIP Networking) and Workshops, Stockholm, Sweden, 12–16 June 2017; pp. 1–2. [CrossRef]
18. Bentstuen, O.I.; Flathagen, J. On Bootstrapping In-Band Control Channels in Software Defined Networks. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
19. Sakic, E.; Avdic, M.; Van Bemten, A.; Kellerer, W. Automated Bootstrapping of A Fault-Resilient In-Band Control Plane. In Proceedings of the Symposium on SDN Research, SOSR '20, Los Angeles, CA, USA, 3 March 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–13. [CrossRef]
20. Park, Y.; Nguyen, D.T.; Kang, B.; Lee, K.; Lee, J.; Choo, H. A Fast Recovery Scheme Based on Detour Planning for In-Band Openflow Networks. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, IMCOM '17, Beppu, Japan, 5–7 January 2017; Association for Computing Machinery: New York, NY, USA, 2017. [CrossRef]
21. Schiff, L.; Schmid, S.; Canini, M. *Medieval: Towards a Self-Stabilizing, Plug & Play, In-Band SDN Control Network*; ACM Sigcomm Symposium on SDN Research (SOSR): San Jose, CA, USA, 2015.
22. Su, Y.L.; Wang, I.C.; Hsu, Y.T.; Wen, C.H.P. FASIC: A Fast-Recovery, Adaptively Spanning In-Band Control Plane in Software-Defined Network. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
23. Asadujjaman, A.S.M.; Rojas, E.; Alam, M.S.; Majumdar, S. Fast Control Channel Recovery for Resilient In-band OpenFlow Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 19–27. [CrossRef]
24. Holzmann, P.; Zitterbart, M. Izzy: A Distributed Routing Protocol for In-band SDN Control Channel Connectivity. In Proceedings of the 2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium), Osnabrueck, Germany, 14–17 October 2019; pp. 18–25. [CrossRef]
25. Silva Freitas, M.; Oliveira, R.; Molinos, D.; Melo, J.; Frosi Rosa, P.; de Oliveira Silva, F. ConForm: In-band Control Plane Formation Protocol to SDN-Based Networks. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 Janaury 2020; pp. 574–579. [CrossRef]
26. Wu, F.; Tian, A. rXstp: A Topology Discovery Mechanism Based on Rapid Spanning Tree for SDN In-Band Control. In Proceedings of the 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), Beijing, China, 14–16 May 2021; pp. 703–706. [CrossRef]
27. An, N.; Lim, H. Poster: Protecting Control Planes in In-Band Software-Defined Wireless Networks. In Proceedings of the 25th Annual International Conference on Mobile Computing and Networking, MobiCom '19, Los Cabos, Mexico, 21–25 October 2019; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
28. Awan, I.I.; Shah, N.; Imran, M.; Shoaib, M.; Saeed, N. An improved mechanism for flow rule installation in-band SDN. *J. Syst. Archit.* **2019**, *96*, 1–19. [CrossRef]
29. Naouri, A.; Wu, H.; Nouri, N.A.; Dhelim, S.; Ning, H. A Novel Framework for Mobile-Edge Computing by Optimizing Task Offloading. *IEEE Internet Things J.* **2021**, *8*, 13065–13076. [CrossRef]

30. Barabási, A.L.; Albert, R. Emergence of Scaling in Random Networks. *Science* **1999**, *286*, 509–512. [CrossRef] [PubMed]
31. Waxman, B.M. Routing of multipoint connections. *IEEE J. Sel. Areas Commun.* **1988**, *6*, 1617–1622. [CrossRef]